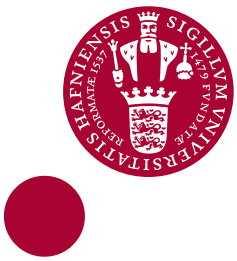

Parallel GPGPU Evaluation of Small Angle X-ray Scattering Profiles in a Markov Chain Monte Carlo Framework

Antonov, Lubomir Dimitrov; Andreetta, Christian; Hamelryck, Thomas Wim

Published in:
Biomedical Engineering Systems and Technologies

DOI:
10.1007/978-3-642-38256-7_15

Publication date:
2013

Document version
Peer reviewed version

Parallel GPGPU Evaluation of Small Angle X-ray Scattering Profiles in a Markov Chain Monte Carlo Framework

Lubomir D. Antonov*, Christian Andreetta*, and Thomas Hamelryck
* These authors contributed equally to this work

The Bioinformatics Section, Department of Biology, University of Copenhagen, Denmark
Corresponding author: thamelry@binf.ku.dk

Abstract. Inference of protein structure from experimental data is of crucial interest in science, medicine and biotechnology. Low-resolution methods, such as small angle X-ray scattering (SAXS), play a major role in investigating important biological questions regarding the structure of proteins in solution.
To infer protein structure from SAXS data, it is necessary to calculate the expected experimental observations given a protein structure, by making use of a so-called forward model. This calculation needs to be performed many times during a conformational search. Therefore, computational efficiency directly determines the complexity of the systems that can be explored.
We present an efficient implementation of the forward model for SAXS with full hardware utilization of Graphics Processor Units (GPUs). The proposed algorithm is orders of magnitude faster than an efficient CPU implementation, and implements a caching procedure employed in the partial forward model evaluations within a Markov chain Monte Carlo framework.

Keywords: SAXS, GPU, GPGPU, MCMC, PROTEIN STRUCTURE DETERMINATION, OPENCL

1 Introduction

Proteins play a crucial role in science, medicine and biotechnology: without them, cellular activities such as catalysis, signaling and regulation would be next to impossible. Protein function is determined by protein structure, which has been proven to be determined by the amino acid sequence [1].

Despite encouraging improvements, determining the ensemble of possible conformations in solution is far from an accomplished goal. High resolution experimental methods, notably X-ray crystallography and Nuclear Magnetic Resonance (NMR), can only partially provide information on such ensembles, and encounter several limitations in fully describing the flexibility of large systems in physiological conditions [2].

Low resolution methods, on the other hand, can more easily provide information on such ensembles. In particular, Small Angle X-ray Scattering (SAXS) provides information on the excess electron density of the sample versus the surrounding environment. Recently, with the advent of automated high-throughput SAXS analysis of

biomolecules [3,4], high-throughput data acquisition is within reach. Since SAXS data only describes the spherical averaging of the electron density of the average conformation of the ensemble, additional information is needed to assist structural interpretation.

Typical *in silico* protein structure determination methods, such as ones based on Markov chain Monte Carlo (MCMC) simulations, propose plausible structural conformations, and compute their associated simulated data by means of a forward model. Then, the simulated and the experimental data are compared using an error model of the experiment. For this procedure to be successful, an efficient method for both sampling protein structures and calculating the simulated data is required.

One approach for the calculation of a SAXS curve from a given structure makes use of the Debye formula [5], which is calculated from a set of spherical scatterers [6,7,8,9]. Another, more recent approach, is based on spherical harmonics expansions [10]. This approach is faster, but becomes problematic for certain structures, such as those with internal cavities [11]. Here, we present an efficient application of the Debye formula, based on a simplified representation of protein structure and the computational power provided by Graphics Processor Units (GPUs).

In recent publications, our group developed probabilistic models for the proposal of protein-like conformations, in full atomic detail, for both backbone and side chains [12,13]. These models were used for the inference of protein structure from NMR data [14]. We also developed a forward model of the scattering profile evaluation, that includes the experimental error associated with SAXS data [15]. The forward model consists of a coarse-grained computation based on the Debye formula. Our main aim is the study of proteins consisting of multiple domains connected by flexible linkers. Such proteins play a major role in the regulation of gene expression, cell growth, cell cycle, metabolic pathways, signal transduction, protein folding and transport [16,17]. With this aim, a computationally efficient forward model for the calculation of SAXS curves is paramount.

We ported our original implementation of the Debye formula to General Purpose computing on Graphics Processing Units (GPGPU). GPUs are parallel computing engines that can offer great advantages in terms of cost-efficiency and low power consumption [18]. One of the emerging standards of choice for their programming is the Open Computing Language (OpenCL), an open standard that provides an abstraction layer over multi- and many-core computational hardware [19]. The OpenCL Debye implementation was utilized as a likelihood term in an MCMC simulation, providing the basis for efficient protein structure determination from low-resolution SAXS data.

2 Methods

2.1 Forward SAXS Computation

The observed data in a SAXS experiment is a one-dimensional intensity curve, $I(q)$, measured at discretized scattering momenta $q = 4\pi\sin(\theta)/\lambda$, with λ the wavelength of the incoming radiation and 2θ the scattering angle between the principal and the ~~probing beam~~ rays. The calculation of a theoretical SAXS profile from a given atomic structure is based on the well-known Debye formula [5]:

$$I(q) = \sum_{i=1}^{M} \sum_{j=1}^{M} F_i(q) F_j(q) \frac{\sin(q \cdot r_{ij})}{q \cdot r_{ij}} \quad (1)$$

where F_i and F_j are the scattering form factors of the individual particles i and j, and r_{ij} is the Euclidean distance between them. The summations run over all the M scattering particles.

2.2 Coarse-grained Protein Models

If some scatterers are fixed relative to each other, they can be approximated by a single large scattering body (*dummy body*). This approximation, more precise at low q, fades with the progression of the scattering angle up to a resolution equal to the scattering diameter of the dummy body. We found that the amino acids constituent to the protein chain can be approximated by one or two large bodies (*dummy atoms*), and that this approximation holds up to scattering angles normally not measured in the current experimental standards [15].

In the two body model, the amino acids are individually represented by two dummy atoms; one representing the backbone, and the other representing the side chain. Glycine and alanine, lacking a side chain with conformational freedom, are represented by a single dummy atom. The dummy atoms are placed at the respective centers of mass (see Fig. 1). A total of 21 form factors need to be estimated for the two body model: one for alanine, one for glycine, one for the generic backbone and 18 for the remaining side chains.

For the one body model, the single dummy atom is placed at the center of mass of the amino acid. Hence, 20 form factors need to be estimated; one for each amino acid type. For a given protein, the one body model allows to represent the molecule with roughly half the number of scattering bodies employed in the two body model. If the experimental data is recorded at low resolutions only, the former is thus clearly preferable for reasons of computational efficiency.

2.3 Form Factor Descriptors

Due to the lack of publicly available high-quality experimental data needed for the estimation of the form factors, artificial data curves were generated for known high-resolution protein structures using the state-of-the-art program CRYSOL [21]. This program computes the theoretical scattering curve from a given full-atom structure using spherical harmonics expansions, therefore limiting its applications at studying compact quasi-globular proteins. We can however use this input to feed a learning protocol, and make use of the Debye formula in eq. 1 to overcome structural assumptions.

Therefore, a large scale Monte Carlo simulation has been conducted to estimate the values of the form factors of the dummy atoms [15]. The resulting profiles for these descriptors are shown in Fig. 2.

In Fig. 3 we show a SAXS curve generated with our method, and the theoretical scattering computed by CRYSOL as a reference.

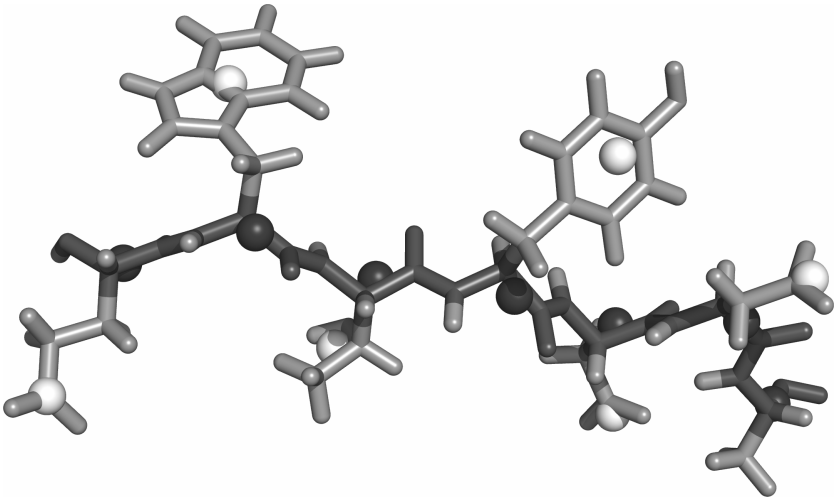

Fig. 1. Coarse-grained models of protein structure. Example of a protein backbone stretch (dark gray) with side chain atoms (light gray). The placement of the dummy bodies for the center of mass of the backbone atoms (dark spheres) and for the side chains (light spheres) are indicated. Figure prepared with PyMOL [20], adapted from [15].

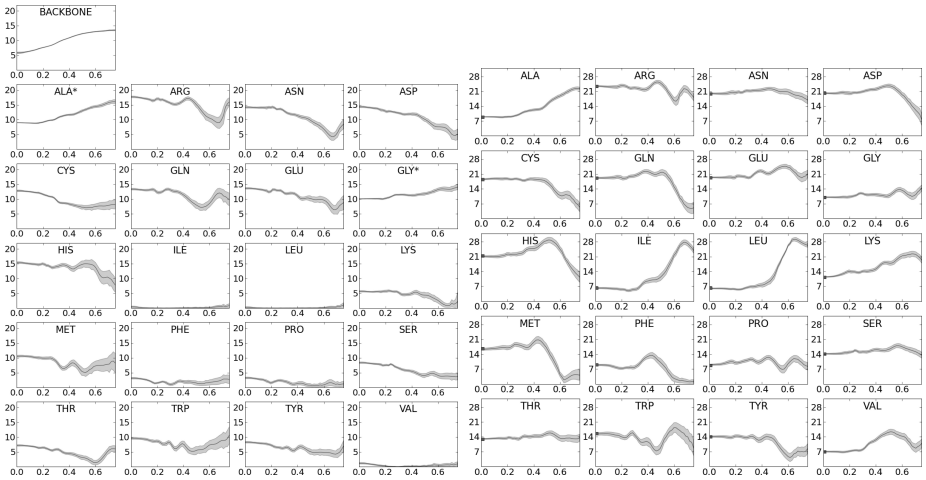

Fig. 2. Form factors. Mean (dark curve) and standard deviations (shaded areas) for the form factors (Y-axis) as a function of q (X-axis). *Left*: backbone and side-chains. An asterisk indicates that this form factor describes both the backbone and side chain atoms of the residue. *Right*: the single body form factors. Figure adapted from [15].

2.4 OpenCL Programming Model

An OpenCL program contains a host program that executes on the CPU, and kernels that execute on the abstracted parallel device. The device consists of one or more compute

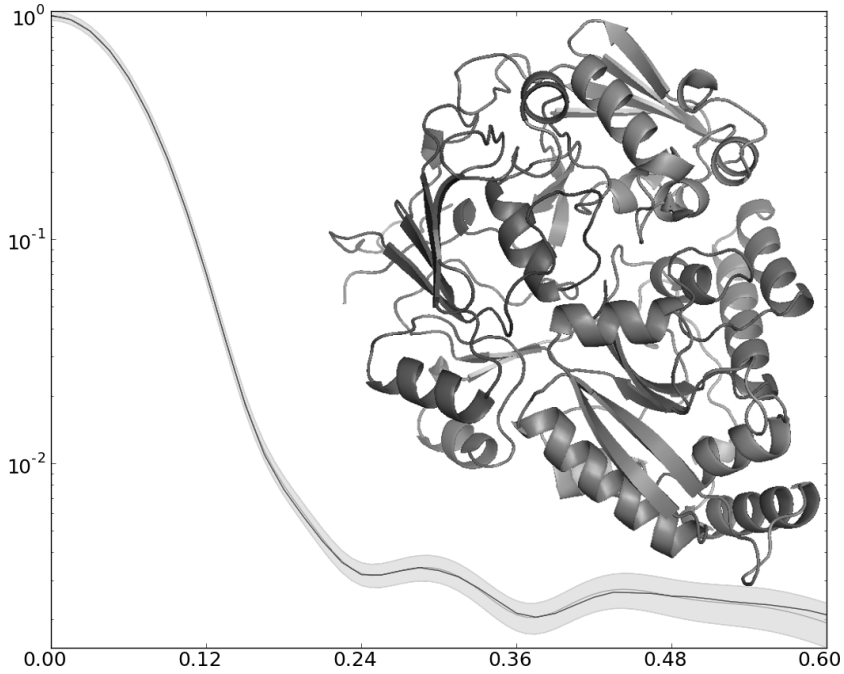

Fig. 3. SAXS profile reconstruction example. Comparison of the reference profiles $I(q)$ computed by CRYSOL from the all atom structure (light gray) and by the two body model (dark line). Error shade indicates the simulated "experimental" error. PDB code 1JET (520 residues). Cartoon made with PyMOL [20].

units, which are composed of one or more processing elements and, in some cases, local memory.

The host program coordinates the execution of the kernels, and can be written in any programming language. Kernels are written in a variant of the latest released C language standard (C99) and are compiled at run time to device-specific instructions. A kernel describes the operations of a single work-item, or thread, and is run simultaneously by a set of work-items called a work-group.

The local memory of a compute unit, if present, is shared by all work-items in a work-group and provides an efficient communication channel among them. It has very low latency and is usually implemented with a full crossbar interface, but is limited in size and does not retain its state between kernel executions.

Kernels execute most efficiently when the size of the work-group matches the size of the compute unit on the OpenCL device and when all work-items in a work-group follow the same execution path.

2.5 Efficient GPGPU Implementation

Parallel Page-Tile SAXS Algorithm. The computation of a SAXS profile is experimentally discretized in a set of q points, and thus naturally provides the first level of

parallelization, into pages. Each page represents the computation of the intensity curve $I(q)$ for a single value of q. A page can be visualized as a square problem matrix of side equal to the number of scattering particles M, with each cell representing the contribution of a single term of the Debye formula for particular i and j.

For performance considerations and direct mapping to the hardware, pages are partitioned into square tiles of side k, where k is set to the specific compute unit size of the OpenCL device. Since each problem matrix is symmetrical, only the tiles encompassing the lower-left triangle and the diagonal are computed and their value is simply duplicated for the mirror tiles in the upper-right triangle of the matrix. The domain decomposition is illustrated in Fig. 4 for an example of 16 scattering particles and work-group size of 4.

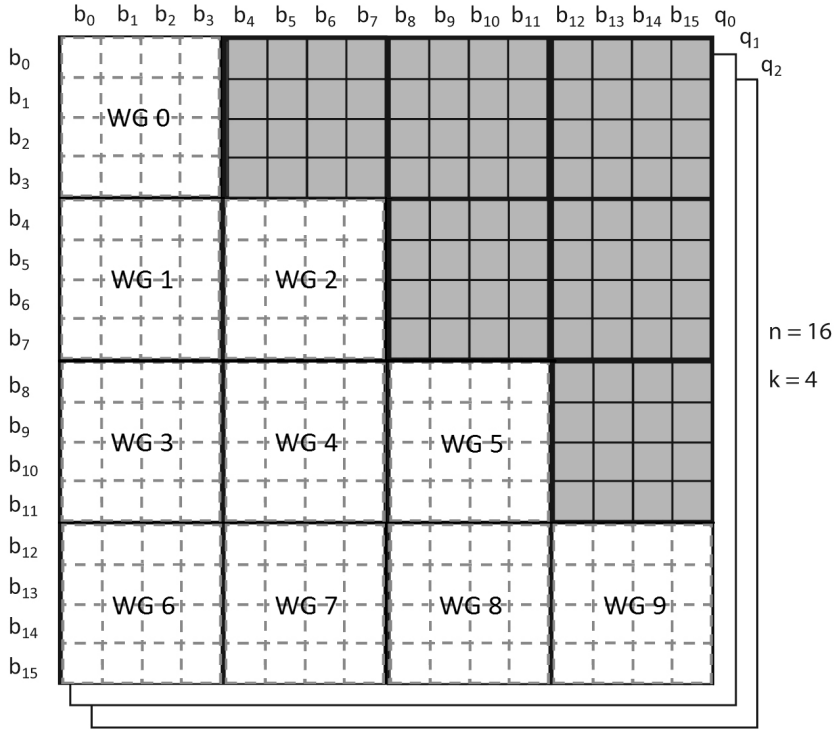

Fig. 4. Domain decomposition for the Page-Tile algorithm. Work-groups operate on square tiles from the matrix. Only tiles in the lower-left part and the diagonal are evaluated.

GPUs suffer performance penalties when they have to work with data that is not aligned to their native architecture. The algorithm therefore pads the data and aligns it to the specified work-group size. The resultant dummy particles participate in the Debye calculations, but they are assigned a form factor of 0, so their contribution to the intensity $I(q)$ is null.

Algorithm 1 Page-Tile SAXS algorithm

Input: scattering momenta, form factors table, scattering particles
Output: intensity curves for the scattering momenta
/*Host program*/
Initialize the parallel algorithm
Transfer input data to GPU global memory and queue the kernels for initial profile calculation
/*Kernels executed on the GPU*/
Map form factors to scattering particles (Kernel 1)
Compute the Debye sum term for each tile (Kernel 2)
Perform vertical tile sum reduction for each page (Kernel 3)
Perform horizontal margin sum reduction for each page to get the intensity curve (Kernel 4)
/*Host program*/
Retrieve the results from GPU global memory

Algorithm 1 presents the pseudocode for the Page-Tile SAXS algorithm. The form factors table, supplied as input, is packed and organized by scattering momentum and particle type. The scattering particles, in addition to their position in three dimensions, have a type in the form of an index into the form factor table. The initial intensity curve calculation comprises four kernels.

In Kernel 1 the form factor table is mapped into a form suited for hardware-efficient parallel access. The form factors are organized by scattering particle, which enables the work-groups with streaming memory access for both center coordinates and form factors.

The majority of execution time is spent in Kernel 2, where the Debye sum terms for the individual tiles are computed. The Debye formula is used for each term, but i and j are limited to the ranges defined by the boundaries of the tile within the global index space. The kernel uses local memory to improve performance, by pre-loading the particles and their form factors, and by performing an in-place parallel reduction to produce the partial sum for the tile. During the Debye calculation, 4x loop unrolling utilizes local registers to further optimize this stage.

Kernel 3 reduces the tile partial sums, which are stored in a global cache, to bottom margin sums that are further reduced by Kernel 4 to yield the final intensity curve.

Tile Recalculation. Markov chain Monte Carlo simulations explore the conformational space of the protein structures by applying partial modifications to an accepted proposal. The average SAXS computation is therefore a partial re-evaluation of a previously computed profile, where only a subset of the bodies changed their position.

It is therefore possible to identify the subset of tiles that needs to be updated. Since the Page-Tile algorithm caches the partial contribution of each tile to the global summation, we can impose a partial recalculation of only the affected tiles (see Fig. 5). This leads to a substantial reduction in the time necessary to derive an intensity curve after a Monte Carlo transition (move).

Algorithm 2 illustrates the pseudocode for tile recalculation. The form factor table from the initial calculation is reused, so execution starts directly with Kernel 5, which

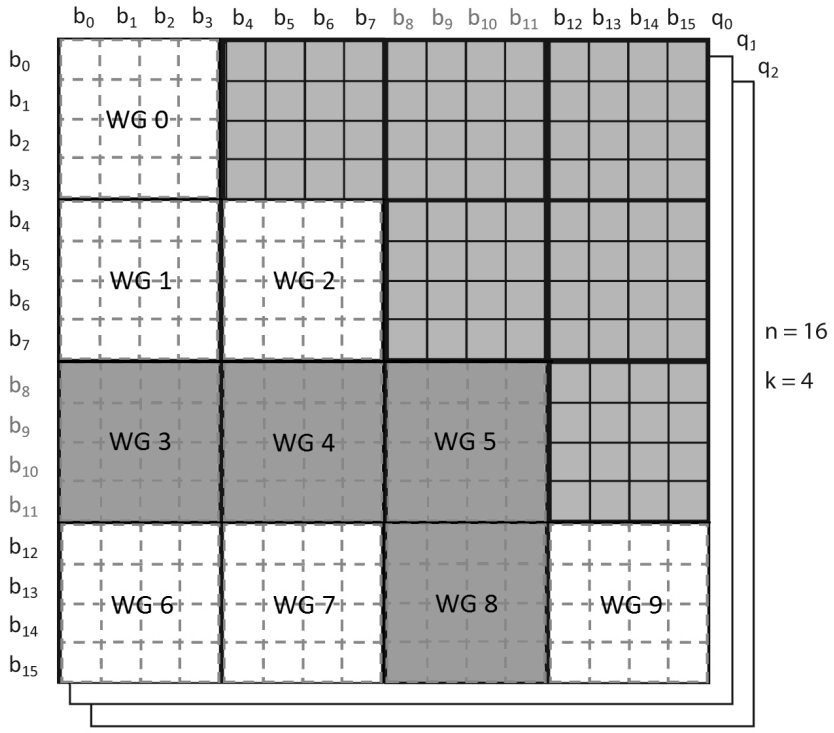

Fig. 5. Problem matrix after a move. Particles b8, b9, b10 and b11 have changed positions. Blocks WG3, WG4, WG5 and WG8 will be recalculated.

identifies the affected tiles and invokes Kernel 2 for them. Kernel 3 and Kernel 4 perform the reductions as in the initial calculation.

Floating Point Precision. Floating point numbers can be stored and manipulated with single precision (SP) or double precision (DP). Mathematical operations on floating point numbers introduce errors, due to the finite precision available. Those errors tend to accumulate when a large number of operations is performed, as is the case with the double sum of the Debye formula. However, the Page-Tile algorithm significantly reduces this error growth, because its successive partitioning of the problem space results in an execution pattern resembling pairwise summation [22]. The algorithm can be executed with SP or DP, paying a performance penalty of a factor of 2 to 4 with DP.

We measured the divergence between the SP and DP executions, and no significant differences arise between the results. Therefore, the SP implementation is used by default.

Algorithm 2 Tile recalculation

Input: moved scattering particles
Output: updated intensity curve for the scattering momenta
/*Host program*/
Transfer input data to the GPU global memory and queue the kernels involved in the profile recalculation
/*Kernels executed on the GPU*/
Compute the Debye sum term for the changed tiles (Kernel 5)
Perform the vertical tile sum reduction for each page (Kernel 3)
Perform horizontal margin sum reduction for each page to get the intensity curve (Kernel 4)
/*Host program*/
Retrieve the results from the GPU global memory

2.6 Monte Carlo Simulation

PHAISTOS is a software framework for protein structure prediction, inference and simulation based on Bayesian principles [23]. PHAISTOS samples protein structures X given the experimental data I_{exp} from a Bayesian posterior distribution $P(X|I_{\text{exp}})$ using an MCMC procedure, similar to [14]. The posterior is given by the formula:

$$P(X|I_{\text{exp}}) \propto P(I_{\text{exp}}|X)P(X) \tag{2}$$

and consists of a prior $P(X)$ that includes probabilistic models of the main and side chains in proteins [24,13], while the likelihood $P(I_{\text{exp}}|X)$ brings in the SAXS data. The likelihood essentially expresses the correspondence between the experimental data and the data calculated from a given structure using the forward model.

For the calculation of the likelihood, we used the error model given in [15]. The resulting likelihood is:

$$P(I_{\text{exp}}|X) = \prod_q \mathcal{N}(I_{\text{exp}}(q)\,|I_{\text{calc}}(q), \sigma(q)), \tag{3}$$

where $\mathcal{N}(\cdot)$ is the normal distribution with mean $I_{\text{calc}}(q)$ and standard deviation $\sigma(q)$, controlled by scaling parameters α and β:

$$\sigma(q) = I_{\text{exp}}(q) \cdot (q+\alpha) \cdot \beta \tag{4}$$

The prior $P(X)$ is brought in indirectly by sampling from the proposal distribution for protein conformations [14].

The majority of time dedicated to each simulation step is spent on computing the energy function for the proposed structure. The GPGPU SAXS algorithm directly reduces this time. Furthermore, at each MCMC step, PHAISTOS performs local moves on a portion of the protein, which allows the Page-Tile algorithm to use the fast tile recalculation path.

2.7 Performance Test Configuration

Performance was measured on a system with a Core i7-920 CPU (4 cores / 8 hardware threads), 12GB of DDR3 RAM and a NVIDIA GeForce GTX 560 Ti GPU with 1GB of GDDR5 RAM. The GTX 560 Ti has 8 compute units with 32 processing elements each, comprising 384 processing elements, with 32KB 32-bit registers and 48KB of local memory for each compute unit.

3 Results and Discussion

3.1 Computational Efficiency of the SAXS Modeling

The Debye formula (Equation 1) leads to a computational complexity of $O(M^2)$, with M the number of scatterers in the structure under examination. Our coarse-grained approach reduces M by representing several atoms by one scattering body (a dummy atom), thereby lowering the complexity to $O\left(\left(M/k\right)^2\right)$, with k the number of scatterers (atoms) described by a dummy body.

The precise value of k is dependent on the primary sequence of the protein. On large datasets, the two dummy model leads to an average k of 4.24 (with a performance increase of $k^2 \simeq 18$). The single body model leads to $k \simeq 7.8$, allowing for a $k^2 \simeq 60$ times faster execution.

3.2 GPGPU Implementation

The performance of the Page-Tile algorithm was measured against a test protein of over a thousand amino acids, modeled with 1888 scattering bodies in the dual dummy atom representation, and a discretization of the q space in 51 scattering momenta. Protein moves were modeled by a random mutation of 40% of the particles, to approximate the asymptotic move rate in a Monte Carlo simulation. The execution times for the model test case are presented in Table 1.

Table 1. Execution times for SAXS curve calculation for a protein with 1888 bodies, 51 scattering momenta and 21 form factors per momentum. Execution times from the top are for a single-core CPU implementation, a parallel GPGPU full computation, and GPGPU partial computation, respectively. Partial computations mimic the costs in a Monte Carlo simulation, where at each step around 40% of the proposal structure is updated.

Algorithm	Time (ms)
CPU SP Time	2408
GPGPU full calculation	9
GPGPU recalculation	6.484

The performance of the algorithm was also measured for protein sizes ranging from 64 to 8192 scattering particles. Each protein was moved 1000 times, in order to obtain

an average of the recalculation steps. Figure 6 shows the speed increase, relative to the CPU single precision implementation, calculated as $t_{\mathrm{cpu}}/t_{\mathrm{gpu}}$.

Figure 6 also illustrates the hardware utilization of the parallel Page-Tile algorithm. The plot shows an asymptotic behavior around problem sizes of 2000 scattering bodies. The GTX 560 Ti GPU employed in the tests is composed of 8 compute units operating on 8 cascading work-groups, allowing for a theoretical peak of 64 active work-groups. The work-group size is 32, therefore the card would reach theoretical peak processing power at 2048 bodies. Our tests show saturation at the same level, thus indicating optimal use of the hardware.

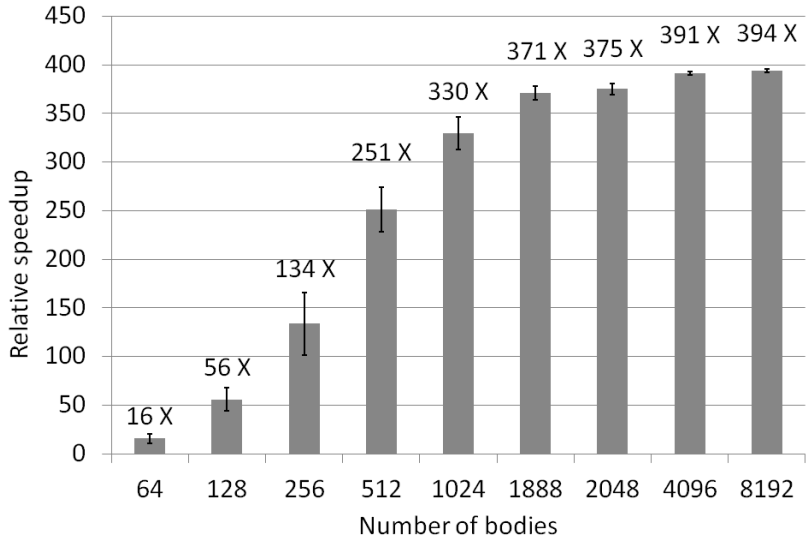

Fig. 6. Algorithm performance for problem sizes ranging from 64 to 8192 scattering bodies, including the model test case of 1888, with error bars showing standard deviation. All SAXS computations involve the summations over 51 scattering momenta. The asymptotic behavior indicates hardware saturation at around 2000 bodies, which is the theoretical maximum for the GPU model used in the tests.

OpenCL is thread-safe and allows access to the same device from multiple processes and threads, so by creating multiple instances of the Page-Tile algorithm, more than one calculation can be run at the same time. This is especially relevant in the case of problem sizes that would not lead to a full GPU saturation, therefore allowing for multi-threaded Monte Carlo simulations.

3.3 Monte Carlo Performance

The effect of the GPGPU SAXS curve calculation on the overall MCMC simulation of the 1888-body test protein was measured in PHAISTOS, by gathering performance

data for 1 million MCMC steps with varying number of concurrent CPU threads, with and without GPU acceleration. The first 100,000 steps were discarded as burn-in and the remaining 900,000 were used for analysis.

The best-performing CPU SAXS algorithm was used in the evaluation. It caches the terms of the Debye formula and uses a lookup table for the sine function, resulting in a five-fold speed increase, at the expense of numerical precision. The MCMC simulation was executed with one to six threads, the maximum possible under the test configuration, due to the significant memory footprint per thread. The lowest execution time was achieved when using four threads, and was used as a basis for comparison with the GPGPU version.

The MCMC simulation, using the GPGPU Page-Tile algorithm, was executed with one to eight CPU threads. Execution time was compared to the multi-threaded CPU version (Fig. 7).

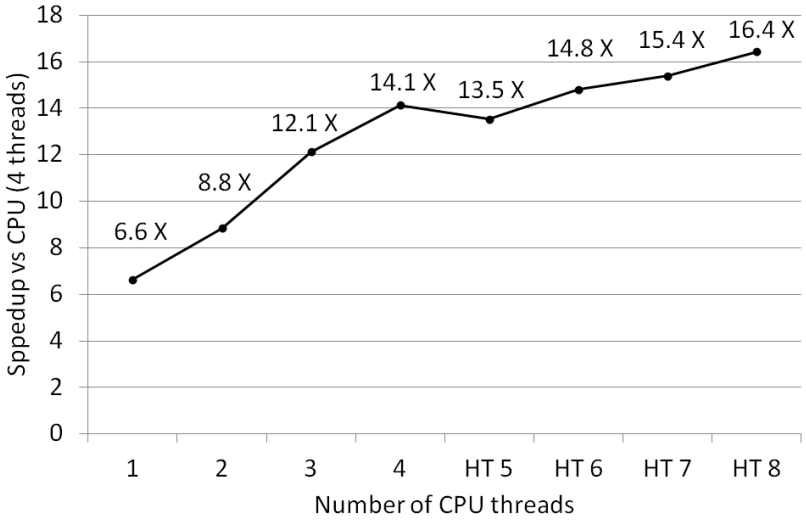

Fig. 7. Relative speedup of PHAISTOS when using the GPGPU SAXS energy term vs. the best-performing CPU configuration (four threads). Threads above four use hyper-threaded CPU cores, so lower performance scaling is expected.

The GPU-accelerated MCMC simulation exhibits consistently better performance compared to the best-performing multi-threaded CPU version. The speed increase scales up with the number of CPU threads, due to two factors: 1) incomplete loading of the GPU by each thread; and 2) concurrent MCMC calculations outside of the SAXS energy term.

When invoked from within a CPU thread, the GPU calculates the SAXS intensity profile and is then idle, while the host thread processes the result and queues a new structure for evaluation. Multiple CPU threads can take advantage of these idle GPU

cycles by queueing additional SAXS curve calculations, thus leading to the performance scaling observed.

While the GPU accelerates the calculation of the SAXS energy term, the CPU host still has to propose a new structure in each MCMC step and to process the result from the energy function calculation. The work done by the CPU limits the speed increase for the overall step. Conversely, running multiple host threads parallelizes these operations, further contributing to the performance scaling.

The theoretical possible speedup under PHAISTOS is 18.6 $\times$, and can be calculated from the Page-Tile algorithm speedup (371 $\times$ from Fig. 6) adjusted for the number of CPU threads (4) and the speedup from using a cache and a sine lookup table (5 $\times$). The observed speed increase of 16.4 $\times$ approaches the theoretical maximum and is clearly limited by the CPU-bound portions of the MCMC simulation.

4 Conclusions

We have presented an efficient implementation of the forward model for the computation of Small Angle X-ray Scattering profiles, utilizing Graphics Processing Units. The application is multi-thread safe, and benchmarks show that the algorithm delivers the full theoretical output of which the hardware is capable.

Parallelization is achieved on multiple levels by taking advantage of the structure of the Debye formula. The first level divides the SAXS evaluation in multiple independent computations according to the binning of the scattering momenta. A nested level then makes full use of the work-groups in the hardware by splitting the inner summation of the Debye formula into separate partial sums. The resulting program runs orders of magnitude faster than an optimized single core CPU implementation.

A caching algorithm on the inner contributions allows for the efficient re-evaluation of SAXS profiles from partially updated structures, delivering even greater performance benefits.

The GPGPU algorithm was integrated into an energy term within the PHAISTOS software framework for protein structure determination, inference and simulation. This yielded a 16-fold speed increase of the Markov chain Monte Carlo simulation, compared to the best multi-threaded CPU implementation, enabling its application to important biological targets. An open source implementation is now available.

Acknowledgements

This research was funded by the Danish Council for Independent Research (FTP, 274-09-0184). CA acknowledges partial funding from the Danish Strategic Research Council (contract number 10-092299), for the HIPERFIT research center (`http://hiperfit.dk`).

Source Code Availability

The source code of our implementation can be found online at: `http://www.phaistos.org` and `http://sourceforge.net/projects/phaistos/files`.

References

1. Anfinsen, C.B.: Principles that govern the folding of protein chains. Science **181**(96) (1973) 223–230
2. Zheng, W., Doniach, S.: Fold recognition aided by constraints from small angle x-ray scattering data. Protein Eng Des Sel **18**(5) (2005) 209–219
3. Toft, K., Vestergaard, B., Nielsen, S., Snakenborg, D.: High-throughput small angle x-ray scattering from proteins in solution using a microfluidic front-end. Anal Chem **80**(10) (2008) 3648–3654
4. Hura, G.L., Menon, A.L., Hammel, M., Rambo, R.P., Poole, F.L., Tsutakawa, S.E., Jenney, F.E., Classen, S., Frankel, K.A., Hopkins, R.C., jae Yang, S., Scott, J.W., Dillard, B.D., Adams, M.W.W., Tainer, J.A.: Robust, high-throughput solution structural analyses by small angle x-ray scattering (saxs). Nat Methods **6**(8) (2009) 606–614
5. Debye, P.: Zerstreuung von rontgenstrahlen. Ann Phys **351**(6) (1915) 809–823
6. Chacon, P., Moran, F., Diaz, J., Pantos, E., Andreu, J.: Low-resolution structures of proteins in solution retrieved from x-ray scattering with a genetic algorithm. Biophys J **74** (1998) 2760–2775
7. Svergun, D., Petoukhov, M., Koch, M.: Determination of domain structure of proteins from x-ray solution scattering. Biophys J **80**(6) (2001) 2946–2953
8. Tjioe, E., Heller, W.: Ornl_sas: software for calculation of small-angle scattering intensities of proteins and protein complexes. J Appl Crystallogr **40** (2007) 782–785
9. Förster, F., Webb, B., Krukenberg, K., Tsuruta, H.: Integration of small-angle x-ray scattering data into structural modeling of proteins and their assemblies. J Mol Biol **382** (2008) 1089–1106
10. Svergun, D.I., Stuhrmann, H.B.: New developments in direct shape determination from small-angle scattering. 1. theory and model calculations. Acta Crystallogr A **47**(6) (Nov 1991) 736–744
11. Koch, M., Vachette, P., Svergun, D.: Small-angle scattering: a view on the properties, structures and structural changes of biological macromolecules in solution. Q Rev Biophys **36**(2) (2003) 147–227
12. Boomsma, W., Mardia, K., Taylor, C., Ferkinghoff-Borg, J., Krogh, A., Hamelryck, T.: A generative, probabilistic model of local protein structure. Proc Natl Acad Sci U S A **105**(26) (2008) 8932–8937
13. Harder, T., Boomsma, W., Paluszewski, M., Frellsen, J., Johansson, K.E., Hamelryck, T.: Beyond rotamers: a generative, probabilistic model of side chains in proteins. BMC Bioinformatics **11** (2010) 306
14. Olsson, S., Boomsma, W., Frellsen, J., Bottaro, S., Harder, T., Ferkinghoff-Borg, J., Hamelryck, T.: Generative probabilistic models extend the scope of inferential structure determination. J Magn Reson **213**(1) (2011) 182 – 186
15. Stovgaard, K., Andreetta, C., Ferkinghoff-Borg, J., Hamelryck, T.: Calculation of accurate small angle X-ray scattering curves from coarse-grained protein models. BMC Bioinformatics **11** (2010) 429
16. Levitt, M.: Nature of the protein universe. Proc Natl Acad Sci U S A **106**(27) (2009) 11079–11084
17. Madl, T., Gabel, F., Sattler, M.: NMR and small-angle scattering-based structural analysis of protein complexes in solution. J Struct Biol (2010) 1–11
18. Göddeke, D., Strzodka, R.a.: Cyclic reduction tridiagonal solvers on GPUs applied to mixed precision multigrid. IEEE Transactions on Parallel and Distributed Systems **22**(1) (2011) 22–32 doi: 10.1109/TPDS.2010.61.

19. Stone, J.E., Gohara, D., Shi, G.: Opencl: A parallel programming standard for heterogeneous computing systems. Computing in Science and Engineering **12** (2010) 66–73
20. Schrödinger, L.: The PyMOL molecular graphics system, version 1.3r1. (2010)
21. Svergun, D., Barberato, C., Koch, M.: Crysol - a program to evaluate x-ray solution scattering of biological macromolecules from atomic coordinates. J Appl Crystallogr **28** (1995) 768–773
22. Higham, N.J.: The accuracy of floating point summation. SIAM J Sci Comput **14** (1993) 783–799
23. Borg, M., Mardia, K., Boomsma, W., Frellsen, J., Harder, T., Stovgaard, K., Ferkinghoff-Borg, J., Røgen, P., Hamelryck, T.: A probabilistic approach to protein structure prediction: PHAISTOS in CASP9. In Gusnanto, A., Mardia, K., Fallaize, C., eds.: LASR2009 - Statistical tools for challenges in bioinformatics, Leeds University Press, Leeds, UK (2009) 65–70
24. Boomsma, W., Mardia, K., Taylor, C., Ferkinghoff-Borg, J., Krogh, A., Hamelryck, T.: A generative, probabilistic model of local protein structure. Proc Natl Acad USA **105** (2008) 8932–8937